

## Laboratorio Nro. 4 Árboles



**Objetivos:** 1. Escoger la estructura de datos apropiada para modelar un problema dado. 2. Solucionar problemas del mundo real con algoritmos.



**Consideraciones:** Lean y verifiquen las consideraciones de entrega,



Leer la Guía



Trabajo en Parejas



Si tienen reclamos,  
regístrenlos en  
<http://bit.ly/2g4TTKf>



Ver calificaciones en Eafit Interactiva



En el GitHub docente, encontrarán la traducción de los Ejercicios en Línea

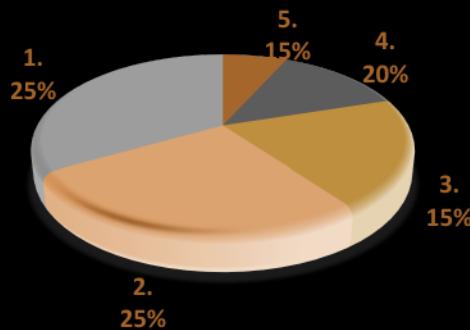


Hoy, plazo de entrega



Subir el **informe pdf** en la carpeta **informe**, el **código del ejercicio 1** en la carpeta **codigo** y el **código del 2** en la carpeta **ejercicioEnLinea**

### Porcentajes y criterios de evaluación



- 1. Simulacro sustentación proyecto
- 2. Análisis de complejidad
- 3. Código de laboratorio
- 4. Simulacro de parcial
- 5. Ejercicio con juez en línea

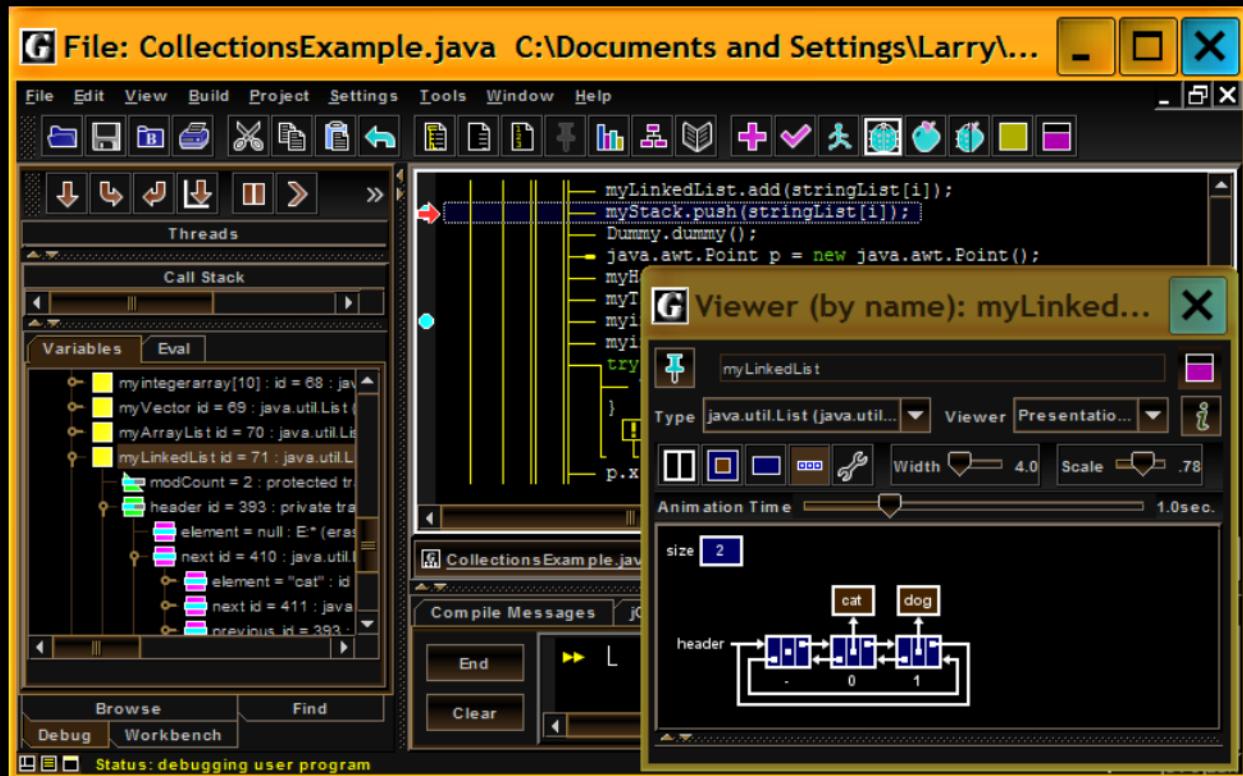
# **1. Simulacro de Proyecto**

**Códigos para entregar en GitHub en la carpeta código**

## Resolver ejercicios

**!** **Nota 1:** Para este taller, de manera especial, se recomienda utilizar el *IDE Jgrasp* (<http://www.jgrasp.org/>) porque tiene un depurador gráfico excelente para estructuras de datos

**!** **Nota 2:** Vean a continuación gráfica de *Jgrasp* para efectos de exemplificación



## 1 Simulacro de Proyecto Códigos para entregar en GitHub en la carpeta código



En la vida real, la documentación de software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



Vean Guía  
numeral 3.4



Código del ejercicio en línea  
en **GitHub**. Vean Guía en  
numeral 4.24



Documentación  
opcional. Si lo  
hacén, utilicen  
**Javadoc** o  
equivalente. No  
suban el HTML a  
GitHub.



*No se reciben* archivos  
en **.RAR** ni en **.ZIP**



Utilicen Java, C++ o Python

**1.1** Los avances en la tecnología han hecho que hoy en día sea necesario almacenar y procesar grandes volúmenes de información. Para poder manejar la información en el disco duro se utiliza un sistema de archivos.

Un sistema de archivos es un componente de un sistema operativo encargado de asignar espacio a los archivos, administrar el espacio libre y permitir el acceso a los datos guardados.



Extraída de [www.google.com](http://www.google.com)

## ESTRUCTURA DE DATOS 1

### Código ST0245

Hoy en día existe la necesidad de almacenar y acceder a un número muy grande de archivos en un sistema de archivos. Una de las funciones del sistema de archivos es listar los contenidos de un directorio eficientemente

El problema de listar el contenido de un directorio consiste en **encontrar eficientemente, los archivos y subdirectorios que se encuentran en un directorio.**

- ▶ El objetivo es desarrollar una estructura de datos para representar los directorios y archivos en un sistema de archivos, y consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio.

 **Ejemplo:** Consideren la siguiente estructura de directorios en un sistema de archivos con el formato [usuario tamaño] archivo o directorio:

Proyecto/

```

├── [root 4.0K] DataSets
|   └── [mauriciotoro 252K] treeEtc.txt
├── [mauriciotoro 4.0K] Plantillas
|   ├── [mauriciotoro 463K] ED1-Guia-Proyecto-Entrega.doc
|   ├── [mauriciotoro 1.5M] ED1-Plantilla-Eafit.pptx
|   ├── [mauriciotoro 255K] ejemplo-de-un-reporte-con-plantilla-ACM.pdf
|   ├── [mauriciotoro 1.1M] plantilla-ACM-en-Latex.zip
|   ├── [root 1.2M] plantilla-ACM-en-Word.doc
|   └── [mauriciotoro 4.0K] SIG Proceedings Template-Jan2015 Zip
        ├── [mauriciotoro 9.4K] acmcopyright.sty
        ├── [mauriciotoro 144K] acm-update.pdf
        ├── [mauriciotoro 336K] flies.eps
        ├── [mauriciotoro 151K] fly.eps
        ├── [mauriciotoro 3.4K] rosette.eps
        ├── [mauriciotoro 65K] sig-alternate-05-2015.cls
        └── [mauriciotoro 124K] sig-alternate-guide.docx

```

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas

Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627

Tel: (+57) (4) 261 95 00 Ext. 9473

## ESTRUCTURA DE DATOS 1

### Código ST0245

```

|   └── [mauriciotoro 572K] sig-alternate-guide.pdf
|   └── [mauriciotoro 255K] sig-alternate-sample.pdf
|   └── [mauriciotoro 26K] sig-alternate-sample.tex
|   └── [mauriciotoro 1.5K] sigproc.bib
└── [mauriciotoro 571K] Proyecto Final ED1 Sistema-directorios Vr 2.0.pdf

```

Si el usuario consulta los subdirectorios y archivos que se encuentran en “*Proyecto/DataSets*” la respuesta debe ser “*treeEtc.txt*”.



**Como otro ejemplo**, si el usuario consulta los archivos y subdirectorios que se encuentran en “*Proyecto/Plantillas/*” y ocupan más de 1 Megabyte, la respuesta debe ser “*ED1-Plantilla-Eafit.pptx*” y “*plantilla-ACM-en-Latex.zip*”.



**Como otro ejemplo** si el usuario consulta los archivos y subdirectorios que se encuentran en “*Proyecto/*” cuyo dueño sea el usuario root, debe retornar “*DataSets/*” y “*plantilla-ACM-en-Word.doc*”

Finalmente, si el usuario consulta por los subdirectorios y archivos que se encuentran en “*Proyecto/Datos3*” la respuesta debe ser “*No such file or directory*”.

### Conjunto de Datos

La estructura de directorios se entrega en un archivo formato TXT con el formato que se muestra en el ejemplo, que es el formato que entrega el programa de la línea de comandos *tree -sha1* que está disponible en Windows, Linux y Mac OS.

En el *github* hay 3 conjuntos de datos: *ejemplito.txt*, *juegos.txt* y *treeEtc.txt*



**[Opc]** Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos.

Con esa información, construyan sus árboles genealógicos en Java usando la clase **BinaryTree**.



**Como un ejemplo**, este sería el árbol genealógico del profesor



En la vida real, los árboles de ancestros son utilizados en Filogenética. Filogenética es la parte de la biología que se ocupa de determinar las relaciones evolutivas entre diferentes grupos de organismos

Si no conocen el nombre de uno de ellos, no lo escriban. No modelen hermanos ni tíos. Modelen en el método `main`, como hijos derechos, los hombres y, como hijos izquierdos, las mujeres. La raíz es “*Mauricio*”.

**1.3 [Opc]** Wilkenson recolectó el árbol familiar de su familia con mucho cariño para dárselo a su abuela Eustaquia. Wilkenson, el nieto de Eustaquia, lo ingresó a Java usando las clases `Node` y `BinaryTree`.

▶ Escriban un método para la clase `BinaryTree` que calcule quién es la abuela materna de una persona. No escriban el método `main`.

! **Nota 1:** El método debe funcionar para cualquier árbol genealógico, no solo para el Wilkenson. Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la abuela materna.

! **Nota 2:** Considere que los ancestros paternos van a la derecha y los ancestros maternos a la izquierda.

## **2. Simulacro de Maratón de Programación sin documentación HTML, en GitHub, en la carpeta ejercicioEnLinea**

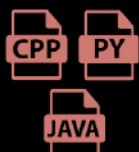
## 2 Simulacro de Maratón de Programación sin documentación HTML, en GitHub, en la carpeta ejercicioEnLínea



Vean Guía numeral 3.3



No se requiere documentación para los ejercicios en línea



Utilicen Java, C++ o Python



**No se reciben** archivos en .PDF ni .TXT



Código del ejercicio en línea en **GitHub**. Vean Guía en numeral 4.24



**Nota:** Si toman la respuesta de alguna fuente, deben referenciar según el tipo de cita. Vean Guía en **numerales 4.16 y 4.17**

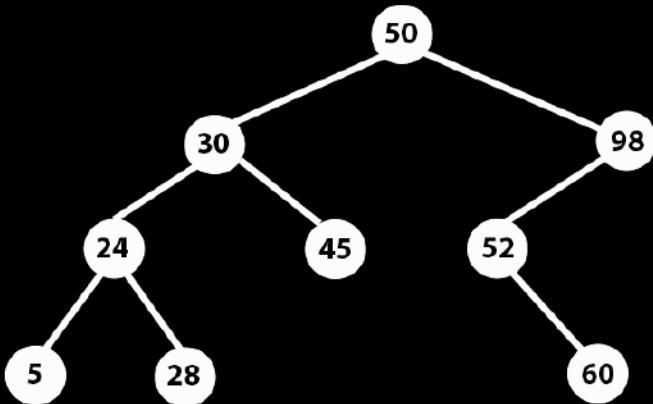


Resuelvan el siguiente ejercicio usando árboles binarios:

Un árbol de búsqueda binario (o árbol binario de búsqueda) es un árbol binario que satisface las siguientes propiedades:

- El subárbol izquierdo de un nodo contiene sólo nodos con valores menores al valor del nodo.
- El subárbol derecho de un nodo contiene sólo nodos con valores mayores al valor del nodo.
- Tanto el subárbol izquierdo como el subárbol derecho tienen que ser también árboles de búsqueda binarios.

**ESTRUCTURA DE DATOS 1**  
Código ST0245



**Figura : Ejemplo de un árbol de búsqueda binario**

El recorrido en pre-orden (Raíz – Izquierda - Derecha) imprime los nodos visitando la raíz del árbol, luego recorriendo el subárbol izquierdo y luego recorriendo el subárbol derecho.

El recorrido en pos-orden (Izquierda – Derecha - Raíz) imprime el subárbol izquierdo primero, luego el subárbol derecho y por último la raíz. Por ejemplo, los resultados del recorrido en pre-orden y pos-orden del árbol de búsqueda binario mostrado en la Figura 1 son:

- **Pre-orden:** 50 30 24 5 28 45 98 52 60

- **Pos-orden:** 5 28 24 45 30 60 52 98 50

► Dado el recorrido en pre-orden de un árbol de búsqueda binario, su tarea es encontrar el recorrido en pos-orden del árbol.



**Entrada**

El valor de cada nodo del árbol de búsqueda binario se dará de acuerdo a un recorrido en pre-orden. Cada nodo tiene un valor que es un entero positivo menor que  $10^6$ . Todos los valores se dan en líneas separadas (un entero por línea). Ud. puede asumir que un árbol de búsqueda binario no contiene más de 10 000 nodos y que no hay nodos duplicados (es decir, el valor de cada nodo es único en el árbol de búsqueda binario dado).

## ESTRUCTURA DE DATOS 1

### Código ST0245



#### Salida

La salida contiene el resultado del recorrido en pos-orden del árbol de búsqueda binario dado en la entrada. Impriman un valor por línea.



#### Ejemplo de entrada

50  
30  
24  
5  
28

45  
98  
52  
60



#### Ejemplo de salida

5  
28  
24  
45  
30  
60  
52  
98  
50



**2.2 [Opc]** Resuelvan el siguiente problema <http://bit.ly/2iNuHaa>

### **3. Simulacro de preguntas de sustentación de Proyecto en la carpeta informe**

### 3 Simulacro de preguntas de sustentación de Proyecto en la carpeta informe



Vean **Guía** numeral 3.4



Exporten y entreguen informe de laboratorio en **PDF, en español o Inglés**



Si hacen el **informe en español**, usen la **plantilla en español**



No apliquen **Normas ICONTEC** para esto

Si hacen el **informe en inglés**, usen a **plantilla en inglés**

#### Ejercicios sobre el Simulacro del Proyecto

3.1 Expliquen qué tipo de árbol utilizaron para representar el sistema de archivos y qué complejidad tiene la operación de búsqueda en él

3.2 **[Opc]** ¿Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?

#### Ejercicios sobre el simulacro de maratón de programación

3.3 Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1 y el 2.2

**ESTRUCTURA DE DATOS 1**  
Código ST0245

**3.4**



Calculen la complejidad del ejercicio realizado en el numeral 2.1 y 2.2, y agregarla al informe PDF

**3.5**



Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.3



**Ejemplos de su respuesta:**

- “n es el número de elementos del arreglo”,
- “V es el número de vértices del grafo”,
- “n es el número de filas de la matriz y m el número de columnas”.

## **4. Simulacro de parcial en informe PDF**

## 4 Simulacro de Parcial en el informe PDF



Para este simulacro,  
agreguen **sus respuestas**  
**en el informe PDF.**



*El día del Parcial no  
tendrán computador,  
JAVA o acceso a internet.*

### 4.1 [Opc] El nodo de un árbol binario se define así:

```
class Nodo {  
    Nodo izq;  
    Nodo der;  
    int dato;  
}
```

Kefo ha dañado a Dayla su código para determinar **cuál es la altura máxima de un árbol binario**. Dayla no recuerda como lo había hecho y les pide ayuda para que lo completen.

```
01 int altura(Nodo raiz) {  
02     if(raiz == null)  
03         return 0;  
04     int izq = _____;  
05     int der = _____;  
06     return Math.max(izq, der); }
```



Completen los espacios en blanco, así:

a) Completen el espacio en línea 04:

---

b) Completen el espacio en línea 05

---

- 4.2 [Opc]** A un árbol binario de búsqueda se le ingresan 9 elementos en este orden: 12, 3, 5, 8, 2, 1, 9, 17.

¿Cuántos nodos hay que recorrer antes de encontrar el número 8? Se cuenta la raíz, pero no se cuenta el nodo donde está el 8.

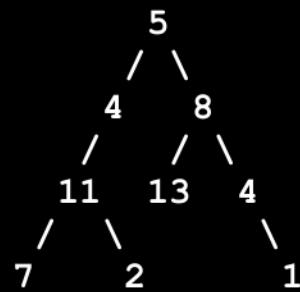
► Elijan la respuesta que consideren acertada:

- a) 2
- b) 1
- c) 3
- d) 5

- 4.3** Definimos el camino desde la raíz hasta una hoja en un árbol binario como una secuencia de nodos empezando en el nodo raíz y bajando hasta una hoja. Una hoja es un nodo que no tiene hijos. Decimos que un árbol vacío no contiene caminos desde la raíz hasta una hoja.



Como un ejemplo, el siguiente árbol tiene 4 caminos desde la raíz hasta una hoja:



Caminos desde la raíz hasta una hoja:

- **Camino 1:** 5 4 11 7
- **Camino 2:** 5 4 11 2
- **Camino 3:** 5 8 13
- **Camino 4:** 5 8 4 1

**ESTRUCTURA DE DATOS 1**  
**Código ST0245**

Para este problema nos interesan las sumas de los elementos en los nodos de esos caminos, por ejemplo, la suma del camino [5,4,11,7] es 27.

Dada la raíz de un árbol binario `Nodo a` y un entero `int suma`, decir si existe un camino desde la raíz hasta una hoja tal que al sumar los valores de los nodos de ese camino la suma sea igual al parámetro `suma`.

Retorne falso si no se puede encontrar un camino con esa condición. Utilice la definición de `Nodo` del punto 1.

Desafortunadamente, al código que hizo Dayla le faltan unas líneas y Kefo está de vacaciones.

```

1 boolean sumaElCamino(Nodo a, int suma) {
2     if (a == null)
3         return _____;
4     if (a.izq == null && a.der == null)
5         return suma == _____;
6     else
7         return sumaElCamino(_____, _____)
8             || sumaElCamino(_____, _____); }
```



**Completen los espacios en blanco, así:**

a) Completen el espacio de la línea 03

---

b) Completen el espacio de la línea 05

---

c) Completen el espacio de la línea 07

---

d) Completen el espacio de la línea 08

---

**4.4 [Opc]** Consideren la siguiente definición de árbol binario:

```
class Node {
    public Node left;
    public Node right;
    public String data;
    public Node(String d) {
        data = d;
    }
}
```

El siguiente algoritmo imprime todos los valores de un árbol en pre orden.

```
01 private void printAUX(Node node) {
02     if (node != null) {
03         System.out.println(node.data);
04         printAUX(node.left);
05         printAUX(node.right);
06     }
07 }
08 public boolean print() {
09     printAUX(root);
10 }
```

**4.4.1** ¿Cuál ecuación de recurrencia que describe el número de instrucciones que ejecuta el algoritmo print en el peor de los casos?

La variable  $n$  representa el número de elementos del árbol.



De acuerdo a lo anterior, elijan la respuesta que consideren acertada:

- a)  $T(n)=T(n-1)+C$
- b)  $T(n)=2.T(n-1)+C$
- c)  $T(n)=2.T(n/2)+C$
- d)  $T(n)=T(n/2)+C$
- e)  $T(n)=T(n+1)+C$

**ESTRUCTURA DE DATOS 1**  
Código ST0245

**4.4.2** ¿Cuál ecuación de recurrencia que describe el número de instrucciones que ejecuta el algoritmo *print* en el peor de los casos?

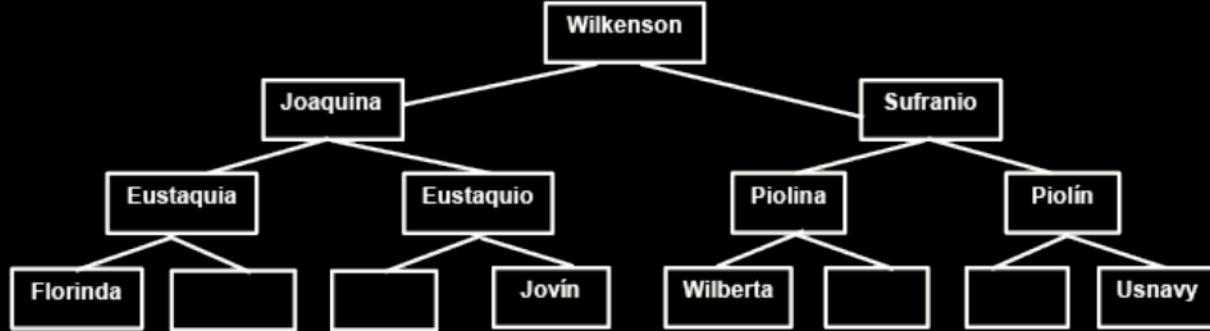
► Elijan la respuesta que consideren acertada:

- a)  $O(n)$
- b)  $O(n^2)$
- c)  $(\log n)$
- d)  $O(n.m)$
- e)  $O(1)$

**4.4.3** ¿Cuál es la salida del algoritmo *print* para el siguiente árbol?

Tengan en cuenta que la raíz es Wilkenson. Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha.

Además que, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.



► Elijan la respuesta que consideren acertada:

- a) Wilkenson, Sufranio, Piolín, Usnavy, Piolina, Wilberta, Joaquina, Eustacio, Florinda, Eustaquia, Yovín
- b) Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda, Wilkenson, Yovín, Eustacio

**ESTRUCTURA DE DATOS 1**  
**Código ST0245**

**c)** Wilkenson, Yovín, Eustaquio, Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda

**d)** Wilkenson, Joaquina, Estaquia, Florinda, Eustaquio, Jovín, Sufranio, Piolina, Wilberta, Piolín, Usnavy

**e)** Sufranio, Piolina, Wilberta, Piolín, Usnavy, Florinda, Wilkenson, Yovín, Eustaquio, Joaquina, Estaquia

**4.4.4** ¿Qué modificación hay que hacer algoritmo print para que arroje la siguiente respuesta para el árbol anterior?:

*Usnavy, Piolín, Wilberta, Piolina, Sufranio, Florinda, Eustaquio, Yovín, Eustaquia, Joaquina, Wilkenson.*

Tengan en cuenta que la raíz es Wilkenson. Como Wilkenson es la raíz, para poder entender el árbol, recomendamos rotarlo 180 grados. Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha

Además, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.



Elijan la respuesta que consideren acertada:

- a)** Cambiar el orden de las líneas 03, 04 y 05 por 05, 04, 03
- b)** Cambiar el orden de las líneas 03, 04 y 05 por 04, 05, 03
- c)** Cambiar el orden de las líneas 03, 04 y 05 por 03, 05, 04
- d)** Cambiar el orden de las líneas 03, 04 por 06, 07
- e)** Intercambiar la línea 03 y la línea 04

**4.5**

**[Opc]** Luis escribió un programa para insertar un número en un árbol binario de búsqueda. En dicho árbol, él quiere tener los números menores o iguales a la raíz a la derecha y los mayores a la raíz a la izquierda.

Ayúdele a completar su código. El algoritmo recibe la raíz de un árbol *p* y un número *a* insertar *toInsert*, y retorna la raíz del árbol con el elemento insertado donde corresponde.

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
 Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
 Tel: (+57) (4) 261 95 00 Ext. 9473

**UNIVERSIDAD**  
**EAFIT**<sup>®</sup>

**AI** Acreditación  
 Institucional  
 Renovación  
 2018 - 2026  
Resolución MEN 2158 de 2018

**ESTRUCTURA DE DATOS 1**  
**Código ST0245**

```

01 private Node insert(Node p, int toInsert) {
02     if (p == null)
03         return new Node(toInsert);
04     if (.....)
05         return p;
06     if (.....)
07         p.left = insert(p.left, toInsert);
08     else
09         p.right = insert(p.right, toInsert);
10     return p;
11 }
```



**Completen los espacios en blanco, así:**

a) Completen, por favor, la línea 4 con la condición que corresponde

---

b) Completen, por favor, la línea 6 con la condición que corresponde

---

**4.6 [Opc]** Dado un árbol *n-ario*, un camino desde la raíz a cualquiera de sus hojas se considera **simple** si la suma de sus elementos pares es igual a la suma de sus elementos impares.

Por ejemplo, un camino desde la raíz hasta una hoja con los elementos [1, 2, 1] es **simple**, pero el camino desde la raíz hasta una hoja [1, 3, 1] **no es simple**.

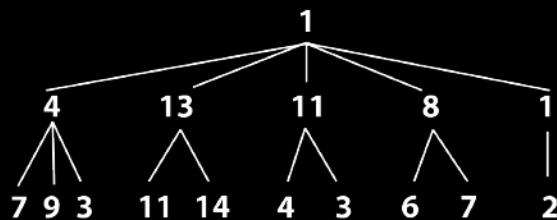
Su tarea es determinar cuántos caminos **simples** hay en un árbol. La implementación de un nodo *n-ario* es la siguiente:

```

class NNodo{
    int val; //Valor en el nodo actual.
    //Hijos del nodo actual.
    LinkedList<NNodo> hijos;
}
```

## ESTRUCTURA DE DATOS 1

Código ST0245



- 4.6.1 ¿Cuántos caminos **simples** hay desde una raíz hasta una hoja en el árbol anterior?

- a. 1
- b. 2
- c. 3
- d. 4

El siguiente código nos ayuda a determinar el número de caminos simples en un árbol, pero faltan algunas líneas. Por favor, complétenlas.

```

01 public int cuantosSimples(NNodo raiz, int suma) {
02     //Arbol vacio
03     if(raiz == null)
04         _____;
05     //Hoja
06     if(raiz.hijos.size() _____) // Si suma es 0
07         return (suma == 0) ? 1 : 0; // Retorne 1, sino, 0
08     int total = 0;
09     for(NNodo n: raiz.hijos)
10         if(n.val % 2 == 0) // Par
11             total += cuantosSimples(n, suma + n.val);
12         else // Impar
13             total += cuantosSimples(n, suma - n.val);
14     return total;
15 }
16
17
18 public int cuantosSimples(NNodo raiz) {
19     int val = (raiz.val % 2 == 0) ?
20             raiz.val : -raiz.val;
21     return cuantosSimples(raiz, val);
22 }
```



**4. 6.2** Completen la línea 4

---

**4. 6.3** Completen la línea 6

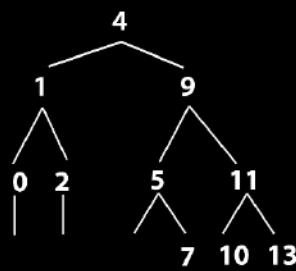
---

- 4.7** Los siguientes algoritmos son el recorrido en in orden y pos orden. En la vida real, estos recorridos son de utilidad para hacer procesamiento del lenguaje natural.

```
void InOrden(Node node) {
    if (node != null) {
        InOrden(node.left);
        System.out.println(node.data);
        InOrden(node.right);
    }
}

void PosOrden(Node node) {
    if (node != null) {
        PosOrden(node.left);
        PosOrden(node.right);
        System.out.println(node.data);
    }
}
```

Consideren el recorrido in orden y pos orden de un **Árbol binario de búsqueda** con los siguientes elementos 4, 9, 1, 5, 7, 11, 13, 2, 0, 10. Ahora, la impresión del recorrido in-orden nos devolverá los elementos, y, la impresión del recorrido pos-orden devolverá.



**4.7.1** ¿Cuál es la impresión pos-orden?

► Elijan la respuesta que consideren acertada:

- a) 0, 2, 1, 7, 5, 10, 13, 11, 9, 4
- b) 0, 1, 2, 4, 5, 7, 9, 10, 11, 13
- c) 0, 2, 1, 7, 10, 5, 13, 11, 9, 4
- d) 4, 1, 0, 2, 9, 5, 7, 11, 13, 10

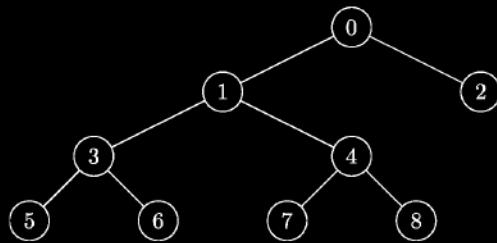
**4.7.2** Supongan que hacemos la comparación elemento a elemento del recorrido en in-orden y el recorrido en pos-orden. ¿Cuántos elementos aparecen en la misma posición en ambos recorridos? Como uno ejemplo, si un recorrido es 1,3,4,2 y el otro es 2,3,4,1, la respuesta es 2 porque el 3 y el 4 aparecen en la misma posición en ambos.

► Elijan la respuesta que consideren acertada:

- a) 7
- b) 2
- c) 5
- d) 8



Nota: Consideren el siguiente árbol para los ejercicios 8,9 y 10



► Elijan la respuesta que consideren acertada:

- a)  $O(n^3)$
- b)  $O(n^2)$ ,
- c)  $O(\log n)$
- d)  $O(n)$

**4.8** Sean  $A$  y  $B$  las salidas de los recorridos pre-orden y pos-orden del árbol binario anterior, respectivamente. Determinen el número de elementos para los cuales se cumple que para  $1 \leq i \leq 8$ .

► Elijan la respuesta que consideren acertada:

- a) 3
- b) 2
- c) 4
- d) 0

**4.9** ¿Cuál es la salida del recorrido in-orden del árbol binario anterior?

► Elijan la respuesta que consideren acertada:

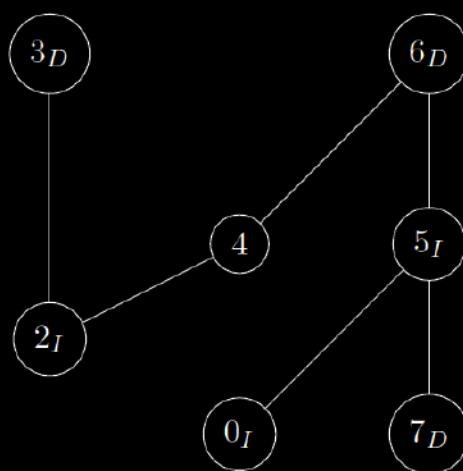
- a) 5, 3, 6, 1, 7, 4, 8, 0, 2
- b) 0, 1, 3, 5, 6, 4, 7, 8, 2
- c) 5, 6, 3, 7, 8, 4, 1, 2, 0
- d) 5, 6, 3, 1, 7, 4, 8, 0, 2

**4.10** ¿Es un **árbol binario de búsqueda** el árbol anterior

► Elijan la respuesta que consideren acertada:

- a) Sí
- b) No

**4.11** El sub-índice D indica que el nodo es un hijo derecho con respecto a su padre. El sub-índice I indica que es un hijo izquierdo con respecto a su padre



**4.11.1** ¿Cuál es la salida del recorrido en *in-orden* del árbol anterior, tomando el nodo 4 como el nodo raíz?

► Elijan la respuesta que consideren acertada:

- a) 3, 2, 0, 7, 5, 6, 4
- b) 2, 3, 4, 0, 5, 7, 6
- c) 4, 2, 3, 6, 0, 5, 7
- d) 4, 7, 3, 6, 2, 0, 5

**4.11.2** Asuma que el nodo 4 es la raíz del árbol binario anterior. Asuma que la salida del recorrido *pre-orden* es  $\{a_1, a_2, a_3, \dots, a_n\}$  y la salida del recorrido *in-orden* es  $\{b_1, b_2, b_3, \dots, b_n\}$ .

¿Cuál es el primer valor de  $i$  para el cual  $a_i = b_i$ ? Note que la  $i$  empieza en 1.

- a) 5
- b) 4
- c) 3
- d) 3



**Por ejemplo,** asuman que las salidas son  $a=\{4,6,5,0,7,2,3\}$  y  $b=\{6,4,7,0,5,2,3\}$ . La respuesta sería 4 porque  $a_4 = b_4 = 0$  e  $i=4$  es el primer valor para el que  $a_i = b_i$ .

**4.11.3** ¿Es un *árbol binario de búsqueda* el árbol anterior?

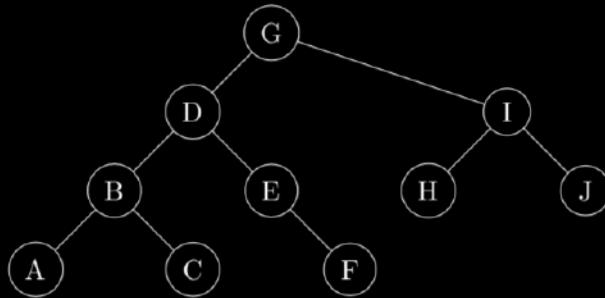


Elian la respuesta que consideren acertada:

- a) Si
- b) No



**4.12** Dada la siguiente lista de enteros, ubica los números en el árbol, de tal manera que el árbol final sea un árbol binario de búsqueda: 7,9,10,8,4,5,6,2,1,3



**4.12.1** ¿Cuáles elementos van en cada letra si se insertan en el orden dado anteriormente?

**ESTRUCTURA DE DATOS 1**  
**Código ST0245**

- i)  $A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8, I = 9, J = 10.$
- ii)  $A = 4, B = 5, C = 6, D = 7, E = 8, F = 9, G = 1, H = 10, I = 2, J = 3.$
- iii)  $A = 7, B = 8, C = 9, D = 10, E = 1, F = 2, G = 3, H = 4, I = 5, J = 6.$
- iv)  $A = 7, B = 8, C = 9, D = 10, E = 1, F = 2, G = 3, H = 4, I = 5, J = 6.$

**4.12.2** ¿Cuál sería el recorrido pre-orden del árbol anterior, asumiendo que siempre se visita primero la hoja de la izquierda?

- a)  $G, D, B, A, C, E, F, I, H, J$
- b)  $A, B, C, D, E, F, G, H, I, J$
- c)  $B, C, D, E, F, G, H, I, J, A$
- d)  $A, C, B, D, I, J, H, E, F, G$

**4.12.3** ¿Cuál es la complejidad asintótica, en el peor de los casos, de encontrar una clave (número) en un árbol binario de búsqueda de  $n$  nodos?

- a)  $O(n)$
- b)  $O(\log(n))$
- c)  $O(n^2)$
- d)  $O(1)$

**4.13** La siguiente es una implementación de árbol n-ario.

```

1  class Nodo {
2      public int id;
3      public int valor;
4      public LinkedList<Nodo> hijos;
5  }
```

Hoy, Liko y Kefa, en clase de Estructura de Datos 1, aprendieron que con los árboles  $n$ -arios se pueden hacer muchísimas operaciones, tales como encontrar el tamaño de un sub-árbol, calcular la suma de los elementos en un sub-árbol, etc. Muchas de estas operaciones son realmente fáciles para Liko; sin embargo, hay una –en especial– difícil para Kefa. Para Kefa es difícil escribir un programa que calcule la suma de los elementos de los sub-árboles. Liko ha escrito un programa para calcular la suma de los elementos en los sub-árboles para enseñarle a Kefa e intencionalmente ha omitido una parte para

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
 Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
 Tel: (+57) (4) 261 95 00 Ext. 9473

**ESTRUCTURA DE DATOS 1**  
**Código ST0245**

asegurarse que, si Kefa entiende el código, él sea capaz de encontrar el error. ¿Podrías ayudarles a Kefa a encontrar el error, por favor? Para propósitos de este ejercicio, Liko se ha asegurado que el árbol dado tiene exactamente  $n$  nodos, y que al método suma se le pasa la raíz del árbol y un arreglo con  $n$  ceros.

```

1 void suma(Nodo raíz, int[] suma) {
2     if(raíz.hijos == null || raíz.hijos.size() == 0) {
3         suma[raíz.id] = raíz.valor;
4         return;
5     }
6     suma[raíz.id] = raíz.valor;
7
8     for(Nodo e: raíz.hijos) {
9         suma(e, suma);
10        suma[_____] = suma[e.id] + suma[raíz.id];
11    }
12 }
```

**4.13.1** Completa, por favor, la línea 10 \_\_\_\_\_

**4.13.2** ¿Cuál es la complejidad asintótica, en el peor de los casos, del código anterior?

- a)  $T(n) = T(n - 1) + c$ , que es  $O(n)$
- b)  $T(n) = 4T(n/2) + c$ , que es  $O(n^2)$
- c)  $T(n) = 2T(n - 1) + c$ , que es  $O(2^n)$
- d)  $T(n) = nT(n - 1) + c$ , que es  $O(n!)$

## **5. [Opcional]**

# **Lecturas Recomendadas**

## 5 [Opc] Lecturas recomendadas



Vean Guía en numeral 3.5 y 4.20



Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**



Si hacen el **informe en español**, usen la **plantilla en español**



No apliquen **Icontec** para esto

Si hacen el **informe en inglés**, usen a **plantilla en inglés**



"El ejercicio de una profesión requiere la adquisición de competencias que se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..." Tomado de <http://bit.ly/2gJKzJD>



Vean Guía en numerales 3.5 y 4.20



Lean a "**Robert Lafore, Data Structures and Algorithms in Java (2<sup>nd</sup> edition), 2002. Chapter 8: Binary Trees.**", y sumen puntos adicionales, así:



Hagan un mapa conceptual con los principales elementos teóricos.

**ESTRUCTURA DE DATOS 1**  
Código ST0245



**Nota:** Si desean una lectura adicional en inglés, consideren la siguiente:  
***“Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2<sup>nd</sup> edition), 2011. Section 6.9 BST”***, que pueden encontrarla en biblioteca

## **6. [Opcional] Trabajo en Equipo y Progreso Gradual**

## 6 [Opc] Trabajo en Equipo y Progreso Gradual



Vean Guía en **numeral 3.5 y 4.20**



Si hacen el **informe en español**, usen la **plantilla en español**

Si hacen **el informe en inglés**, usen a **plantilla en inglés**



Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**



No apliquen **Normas Icontec** para esto



**El trabajo en equipo es una exigencia del mercado.** "Mientras algunos medios retratan la programación como un trabajo solitario, pero requiere mucha comunicación y trabajo grupal. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques bien con otras personas" Tomado de <http://bit.ly/1B6hUDp>



Vean Guía en numerales 3.6, 4.21, 4.22, 4.23



▶ Entreguen copia de todas las actas de reunión usando el *tablero Kanban*, con fecha, hora e integrantes que participaron



▶ Entreguen el reporte de *git* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron



▶ Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares



**Nota:** Estas respuestas también deben incluirse en el informe PDF

**7. [Opcional]**

**Laboratorio en Inglés con  
plantilla en Inglés**

## 7 [Opc] Laboratorio en inglés



Vean Guía en  
**numeral 3.5 y 4.20**



Si hacen el **informe  
en español**, usen la  
plantilla en español

Si hacen el **informe  
en inglés**, usen a  
plantilla en inglés



Exportar y entregar informe  
de laboratorio en **PDF, en  
español o Inglés**



No apliquen **Normas  
Icontec** para esto



**El inglés es un idioma importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo.**

**Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados. Tomado de [goo.gl/4s3LmZ](http://goo.gl/4s3LmZ)**



Entreguen el código y el informe en inglés.

## Resumen de Ejercicios a Resolver

**1.1** Desarrollar una estructura de datos para representar los directorios y archivos en un sistema de archivos, y consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio.

**1.2 [Ejercicio Opcional]** Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos. Con esa información, construyan sus árboles genealógicos en Java usando la clase `BinaryTree`

**1.3 [Ejercicio opcional]** Escriban un método para la clase `BinaryTree` que calcule quién es la abuela materna de una persona. No escriban el método `main`.

**2.1** Resuelvan el problema usando árboles binarios

**2.2 [Ejercicio Opcional]** Resuelvan el siguiente problema <http://bit.ly/2iNuHaa>

**3.1** Expliquen qué tipo de árbol utilizaron para representar el sistema de archivos y qué complejidad tiene la operación de búsqueda en él.

**3.2 [Ejercicio Opcional]** Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?

**3.3** Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1 y el 2.2

**3.4** Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1 y el 2.2

**3.5** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.3

### 4. Simulacro parcial

**5. [Ejercicio Opcional]** Lectura recomendada

**6. [Ejercicio Opcional]** Trabajo en Equipo y Progreso Gradual

**7. [Ejercicio Opcional]** Laboratorio en inglés

# **Ayudas para resolver los Ejercicios**

<b>Ayudas para el Ejercicio 1.....</b>	<b><u>Pág. 38</u></b>
<b>Ayudas para el Ejercicio 1.3.....</b>	<b><u>Pág. 38</u></b>
<b>Ayudas para el Ejercicio 2.1.....</b>	<b><u>Pág. 39</u></b>
<b>Ayudas para el Ejercicio 2.2.....</b>	<b><u>Pág. 39</u></b>
<b>Ayudas para el Ejercicio 3.2.....</b>	<b><u>Pág. 39</u></b>
<b>Ayudas para el Ejercicio 3.4.....</b>	<b><u>Pág. 39</u></b>
<b>Ayudas para el Ejercicio 4.....</b>	<b><u>Pág. 40</u></b>
<b>Ayudas para el Ejercicio 5.....</b>	<b><u>Pág. 40</u></b>
<b>Ayudas para el Ejercicio 6.1.....</b>	<b><u>Pág. 40</u></b>
<b>Ayudas para el Ejercicio 6.2.....</b>	<b><u>Pág. 40</u></b>
<b>Ayudas para el Ejercicio 6.3.....</b>	<b><u>Pág. 40</u></b>



## Ayudas para el Ejercicio 1



**Pista:** Si deciden hacer la documentación, consulten la *Guía* en numeral 4.1



## Ayudas para el Ejercicio 1.3



**Como un ejemplo**, en el árbol genealógico de Wilkenson, Eustaquia es la abuela materna de Wilkenson. Otro ejemplo, Wilberta es la abuela materna de Sufranio y no se conoce la abuela materna de Eustaquia

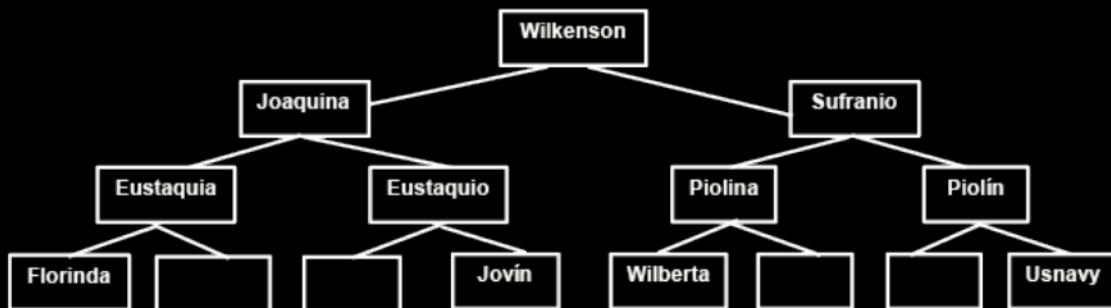


**Pista 1:** Utilicen el siguiente estereotipo para el método:

```
public String getGrandMothersName(String name)
```



**Pista 2:** Retornen una cadena vacía si no se encuentra el nombre de la persona *name* o la abuela materna de *name*. A continuación, un ejemplo de un árbol genealógico





## Ayudas para el Ejercicio 2.1



**Pista 1:** Lean sobre <http://bit.ly/1V5RqKL>



**Pista 2:** No es posible reconstruir un árbol binario, en general, con un el recorrido pre-orden porque habría muchas posibles respuestas. Para poder hacer este ejercicio hay que recordar que es un árbol binario de búsqueda, es decir, donde los menores van a la izquierda y los mayores van a la derecha.



## Ayudas para el Ejercicio 2.2



**Pista:** Use *búsqueda en profundidad (pre orden, in orden o pos orden)*. Es similar a lo que hacen en Lenguajes de Programación para construir el árbol sintáctico en una expresión aritmética con paréntesis.



## Ayudas para el Ejercicio 3.2



**Pista 1:** Lean sobre <http://bit.ly/1V5RqKL>



**Pista 2:** Lean sobre <http://bit.ly/2i8DgM4>



## Ayudas para el Ejercicio 3.4



**Pista 1:** Vean Guía en numeral 4.11



**Pista 2:** Vean Guía en numeral 4.19



## Ayudas para el Ejercicio 4



**Pista 1:** Vean Guía en numeral 4.18



## Ayudas para el Ejercicio 5



**Pista 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacoo.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>



## Ayudas para el Ejercicio 6.1



**Pista 1:** Vean Guía en numeral 4.21



## Ayudas para el Ejercicio 6.2



**Pista 1:** Vean Guía en numeral 4.23



## Ayudas para el Ejercicio 6.3



**Pista 1:** Vean Guía en numeral 4.22

# ¿Alguna inquietud?

## CONTACTO

Docente Mauricio Toro Bermúdez  
Teléfono: (+57) (4) 261 95 00 Ext. 9473  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)  
Oficina: 19- 627

Agenden una cita dando clic en la pestaña  
-Semana- de <http://bit.ly/2gzVg10>